An instrumentation interface application that obtains information from an instrument directory and generates a graphic user interface populated based on the information. The information for a given instrument may be formatted in a manner so as to allow the interface application to interpret the information to be displayed in a certain manner. By delegating the such details of the instrument information to a directory instead of the interface application, the interface application can be utilized more effectively for relatively large biological laboratories that have many instruments of different types. The graphic user interface generated in such a manner can be used to monitor and control a variety of different aspects of the laboratory, including instrument management, sample management, and study-related management.
FIG. 1
FIG. 2-4

Instrument Directory

Instrument Information
- Instrument "A"
  - Physical location = "aaaa"
  - Instrument type = "λ"
  - Instrument group = "bbbb"
- Publishing information:
  - Event topic = "cccc"
  - Status topic = "dddd"
  - Publisher = "eeee"
- Service Providers
  - Service provider α
    - Location = 1111
    - Interfaces: Instrument, Service History, ...
  - Service Provider β
    - Location = 2222
    - Interfaces: Status Services, Container Provider

Active Instruments
- Instrument "A"
- Instrument "B"
- Instrument "C"

Instrument Information
- Type "λ"
  - Location = 3333
- Type "α"
  - Location = 4444

Storage

Naming and Directory Interface

Network
FIG. 3
FIG. 4C
Instrument Directory

**Instrument Information**

**Instrument ID:** "A" "Array"

**Available monitor parameters:**
1. Instrument ID
2. Run ID
3. Instrument State
4. Instrument Substate
5. Instrument Startup
6. Laser On/Off
7. EP On/Off
8. Lid Open/Closed
9. Array Serial Number
10. Array Usage
11. Laser Intensity
12. Laser Position
13. Time Remaining
14. Total Time
17. EPC Temp. [degC 0 100 EPC_T_actual EPC_T_nominal]
18. Cuvette Temp. [degC 0 100 Cuvette_T_actual Cuvette_T_nominal]
19. Laser Power [mW 0 45 Laser_P_actual Laser_P_set]
20. Laser Current [A 0 15 Laser_I_actual Laser_I_set]
21. Events
22. Errors

**Available control parameters:**
1. Laser [mW 0 25 Value]
1a. Set Power
1b. Apply Power
1c. Shut down
2. Comments

**FIG. 5A**
FIG. 7B
FIG. 8B
Start

Receive request for an user interface

Obtain information associated with the user interface

Generate a GUI populated with the information obtained

Stop

FIG. 10
FIG. 14
FIG. 16
FIG. 17
FIG. 21
SYSTEM AND METHOD FOR GENERATING USER INTERFACES FOR DIFFERENT INSTRUMENT TYPES

CLAIM OF PRIORITY

[0001] This U.S. patent application claims priority to U.S. Provisional Patent Application No. 60/386,296 entitled “Informatics System Architecture” filed Jun. 4, 2002 and U.S. Provisional Patent Application No. 60/411,574 entitled “Integration Instructions For Informatics System Architecture” filed Sep. 16, 2002 which are hereby incorporated by reference in their entirety; attorney docket number ABIOS.044A entitled “System And Method For Open Control And Monitoring Of Biological Instruments”, attorney docket number ABIOS.045A entitled “System And Method For Discovery Of Biological Instruments”, and attorney docket number ABIOS.046A entitled “System And Method For Providing A Standardized State Interface For Instrumentation”.

BACKGROUND

[0002] 1. Field

[0003] The present teachings generally relate to a system and methods for interacting with components used in laboratory environment and more particularly to a coordination schema and dynamically generated user interface for monitoring, coordinating, and controlling components and instrumentation.

[0004] 2. Description of the Related Art

[0005] Biological laboratories, in particular those focused on genetic and molecular biological work, generally employ a wide variety of instrumentation and applications, the activities of which must be coordinated in order to conduct experiments and analysis. A selected analysis may require the use of numerous instruments including, for example, an electrophoresis component, a thermalcycler, a mass spectrometer, and/or a gene sequencer. To further complicate matters, some assays and high throughput approaches require the manipulation of hundreds, if not thousands, of samples, each of which must be processed through a multiplicity of tests, the collected data from which must then be analyzed by secondary analytical instrumentation and applications before generating the desired result. In addition, the physical and logistical demands of the laboratory increasingly utilize robotics as well as human investigators in order to locate and deliver samples to the proper instrumentation.

[0006] Communication complexity in the laboratory environment represents another area of potential inefficiency where problems may arise. Each instrument, robot, hardware and/or software application generally has input and output parameters/information which must be properly configured/accessed in order to be utilized. This information may include machine or application specific experimental parameters, commands, and/or catalog information. A particular limitation of many conventional systems is that not all biological instruments/applications are designed to communicate readily with each other and, as a consequence, human intervention is frequently required to coordinate the activities of each instrument and application. In many instances, a centralized information flow control system would be desirable in order to reduce human intervention and errors. However, because a given laboratory environment may contain many different instruments/applications, produced by different manufacturers with different requirements and input/output data types, it is difficult to integrate conventional instrument platforms into a unified system. Furthermore, as new instruments/applications are introduced into the laboratory environment with existing systems, additional difficulties may be encountered arising from configuration limitations associated with previous laboratory setups. A still further drawback found in many conventional systems is that a common user interface for monitoring or controlling each of the instruments/applications in the laboratory environment is not available. As a consequence, certain instruments/applications must be monitored and controlled separately from other laboratory components resulting in a less desirable decentralized system architecture.

SUMMARY

[0007] One aspect of the present teachings relates to a system for providing a centralized user interface for a biological laboratory having a plurality of biological processing instruments adapted to process a plurality of biological samples. The system comprises an information component for instruments that includes information about the plurality of the biological processing instruments. The information for a given instrument comprises a list of the instrument’s parameters that can be monitored and a list of the instrument’s parameters that can be controlled. The system further comprises an interface application that generates the graphic user interface based on the information for one or more of the instruments. The interface thus generated is populated according to at least a portion of the instrument’s monitor and control parameters thereby allowing the interface application to generate different interfaces based on a single adaptable template interface.

[0008] In certain embodiments, the centralized user interface for a given instrument allows a user to monitor and control the instrument. In one of such embodiment, the interface application uses the information about the plurality of instruments to generate an instrument management graphic user interface that lists instruments available for use.

[0009] In certain embodiments, the system further comprises an information component for samples that includes information about the plurality of biological samples. The samples’ information component allows the interface application to generate adaptable user interfaces based on the samples’ information. In one of such embodiments, the interface application uses the information about the plurality of biological samples to generate a sample management user interface that lists the samples in the biological laboratory. In one of such embodiments, the sample and instrument information are combined to allow the interface application to generate a study management user interface. Such user interface is adapted to allow a user to plan a manner in which data is taken during a run.

[0010] In certain embodiments, the system further comprises an information component for an analysis module that analyzes the data from one or more biological processing instruments. The analysis module’s information component allows the interface application to generate adaptable graphic user interfaces based for viewing results of the analyzed data.
In certain embodiments, the plurality of biological processing instruments includes any combination of a sample storage device, a sample transferring robotics device, a thermocycler device, a mass spectrometer, a sequencer device, an electrophoresis device, an array device, an analysis computing device, and a data storage device. In certain embodiments, the instrument component is part of an instrument directory.

Another aspect of the present teachings relates to a method for generating a centralized user interface for a biological laboratory having a plurality of biological processing instruments adapted to process a plurality of biological samples. The method comprises receiving a request for a selected user interface; obtaining information associated with the requested user interface; and generating the requested user interface by populating a template user interface based on the information obtained.

In certain implementations of the method, receiving the request comprises receiving a request for a user interface for one or more biological processing instruments. In one of such implementations, obtaining information comprises obtaining information about the one or more biological processing instruments from an instrument directory.

In certain implementations of the method, receiving the request comprises receiving a request for a user interface for one or more biological samples. In one of such implementations, obtaining information comprises obtaining information about the one or more biological samples from a sample list.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 illustrates an exemplary biological laboratory having a plurality of biological processing instruments that can be monitored and controlled by a user interface;

FIGS. 2-1 to 2-5 illustrate how various instruments can be integrated so as to be recognized by a control system;

FIG. 3 illustrates various functional features of the biological laboratory that can be represented by the user interface;

FIG. 4A illustrates a graphic user interface (GUI) application that generates a GUI for a particular instrument based on information about that instrument, wherein the generated GUI is based on a common template GUI;

FIG. 4B illustrates a web browser that could be used as a user interface;

FIG. 4C illustrates a dynamic generation of an instrument-related GUI when a new instrument is added to the system;

FIG. 5A illustrates an exemplary instrument information that can be used for generating the GUI for the exemplary instrument;

FIG. 5B illustrates an exemplary GUI for monitoring the instrument of FIG. 5A;

FIG. 5C illustrates an exemplary GUI for controlling the instrument of FIG. 5A;

FIG. 6 illustrates an exemplary GUI for monitoring a list of samples;

FIGS. 7A-C illustrate exemplary GUIs that can be generated from instrument and/or sample information, thereby allowing representations of instrument and/or study management features;

FIGS. 8A-B illustrate exemplary GUIs that can be generated to display either raw data as they come out of instrument(s);

FIG. 9 illustrates an exemplary GUI that can be generated to display an analyzed data;

FIG. 10 illustrates a process that generates a GUI based on the available information;

FIG. 11 illustrates a communication sequence and functionalities of a dynamic container creation service;

FIG. 12 illustrates a sample snapshot from a data collection instrument;

FIG. 13 illustrates a custom client that may use an exemplary function subscribeInstrumentEvents() to prepare itself to receive information from exemplary ICFSstatusEvents() and ICFContainerStatusEvents() functions;

FIG. 14 illustrates an exemplary flowchart that describes one possible implementation of a pulling and pushing process;

FIG. 15 illustrates how a custom client can "push" container information to a service provider to persist in the service provider's database allowing it to be run at a later time;

FIGS. 16-20 illustrate various embodiments of XML schema for application and container definitions that may be used in implementation of service sets and container creation/utilization services; and

FIG. 21 illustrates one method for implementing a web-browser based dynamically loaded user interface implemented using the JAVA language.

DETAILED DESCRIPTION OF CERTAIN EMBODIMENTS

These and other aspects, advantages, and novel features of the present teachings will become apparent upon reading the following detailed description and upon reference to the accompanying drawings. In the drawings, similar elements have similar reference numerals.

One aspect of the present teachings relates to a user interface system (UI) that provides means for a user to interact with a plurality of instruments and applications. The user interface system further facilitates monitoring and controlling of the instruments/applications in a centralized and a user-friendly graphical environment. FIG. 1 illustrates a functional block diagram showing a system for biological analysis 1000 wherein an application programming interface (API) 1032 provides communication functionality between a user interface 1034 and several possible types of instruments 1002 associated with biological analysis. Such instruments may include, by way example, one or more sample storage devices 1010; one or more sample transfer robotics devices 1012; one or more sample multiplexing devices (e.g., a thermocycler 1014); one or more sample analyzing devices such as a mass spectrometer 1016, a sequencer 1020, an electrophoresis device 1022, an array analysis
device 1024; one or more analysis computing devices 1026; and one or more data storage devices 1030. In one aspect, the interface 1032 is configured to generate a user interface (UI) 1034 that facilitates the user’s interaction with the exemplary instrument(s) described above. More specific examples of these instruments are described below in the form of examples.

[0038] It will be appreciated that any type of instruments not illustrated in FIG. 1 may be included in the analytical system without departing from the spirit of the present teachings. Moreover, the presence of the type(s) of instruments as illustrated in FIG. 1 does not necessarily mean that such a instrument is required in the analytical system.

[0039] In one aspect, the interface 1032 advantageously can be configured to provide a common graphical user interface (GUI) that dynamically adapts to the various parameters associated with the instruments. The interface 1032 may obtain the parameters for a given instrument in a manner described below. By delegating the details about the instruments to some database, the GUI formation may be greatly simplified since the user does not need to worry about the specifics of the instruments in configuring the application that generates the GUI.

[0040] In one aspect, dynamic generation of the user interface is accomplished in such a manner so as to simplify its creation by utilizing database-stored instrument and application definitions detailing input/output requirements for each type or class of instrument and application. This information is automatically associated with the appropriate field of the user interface thereby facilitating its generation and freeing the user from having to know extensive details of the I/O requirements of each instrument/application while still allowing for relatively simple construction/configuration of the user interface.

[0041] FIG. 1 also illustrates a sample 1004 being processed by the aforementioned exemplary instruments. The sample may be identified in any number of ways, including for example, a barcode or other unique identifier 1006. As described below in greater detail, a plurality of such samples may be managed via the interface 1032 in conjunction with the plurality of exemplary instruments.

[0042] Furthermore, FIG. 1 also depicts the exemplary instruments having a physical portion and a programmable portion. The programmable portion may represent, for example, one or more processes that may be running in the instrument. The physical portion may represent the actual hardware portion of the instrument. As FIG. 1 illustrates, the programmable portions of the analysis-related instruments (exemplified as the thermocycler 1014, mass spectrometer 1016, sequence 1020, electrophoresis 1022, and array 1024) may also be functionally coupled to the analysis computing device 1026 such that data can be passed therebetween and be processed remotely from the instrumentation. The computer 1026 may be functionally connected to the data storage devices 1030 so as to allow the computer 1026 to store and/or retrieve data therefrom. In certain embodiments, the data storage device 1030 may also be accessed by the instruments directly so as to store and/or retrieve data without requisite interaction by the computer 1026. The instruments and computing-related components arranged in the foregoing manners are representative of various configurations which allow biological measurements and analysis to be conducted using a centralized coordination system through which an investigator interacts using the user interface.

[0043] In general, it will be appreciated that the processors may comprise, by way of example, computers, program logic, or other substrate configurations representing data and instructions, which operate as described herein. In other embodiments, the processors can comprise control circuitry, processor circuitry, processors, general purpose single-chip or multi-chip microprocessors, digital signal processors, embedded microprocessors, microcontrollers and the like.

[0044] Furthermore, it will be appreciated that in one embodiment, the program logic may advantageously be implemented as one or more components. The components may advantageously be configured to execute on one or more processors. The components include, but are not limited to, software or hardware components, modules such as software modules, object-oriented software components, class components and task components, processes methods, functions, attributes, procedures, subroutines, segments of program code, drivers, firmware, microcode, circuitry, data, databases, data structures, tables, arrays, and variables.

[0045] FIGS. 2-1 to 2-5 illustrate a manner in which various devices can be integrated into a laboratory system. Such an advantageous manner is described in greater detail in a pending application titled “System and Method for Open Control and Monitoring of Biological Instruments,” attorney dock number ABIOS.044A, which is hereby incorporated by reference in its entirety.

[0046] FIG. 2-1 illustrates an integration framework of a biological laboratory system having a plurality of exemplary devices indicated as a robot 2-130, an electrophoresis device 2-134, a sample amplification component 2-136, a mass spectrometer 2-138, an analysis computer 2-142, and a thermocycler 2-144. Such devices may be connected to a common network, and be managed by a management system 2-104 via the common network. The management of the devices may be facilitated by an instrument directory 2-112 that includes information about the devices. Such information may include the types of interfaces supported by the devices, device-specific information such as serial number and physical location, and identifiers that allow communication via the network. In certain embodiments, the instrument directory 2-112 also includes a list of updated devices that are available for use.

[0047] The management of the devices may also be facilitated by an instrument-type server 2-114. In certain embodiments, the instrument-type server 2-114 provides the management system 2-104 with information about various devices’ requirements so that the management system 2-104 can monitor and control the devices in an appropriate manner.

[0048] The management of the devices may also be facilitated by a messaging service 2-118. The messaging service 2-118 allows communication between the various management-related entities (e.g., management system, instrument directory, instrument-type server, and other entities described below), various devices, and other entities in the laboratory system. In certain embodiments, the messaging service 2-118 provides such communication without having to rely on a direct communication between components.
The management of the devices may also be facilitated by an instrument control component 2-122. In certain embodiments, the instrument control component 2-122 comprises a computing device configured to run service provider software or applications 2-123. Such software may be configured to provide control over the devices as well as provide devices’ status updates to the management system 2-104. In certain embodiments, the service provider software 2-123 may implement one or more software interfaces that allow communication between the management-related components with the various devices in a meaningful manner.

The management of certain devices in the laboratory system may also be facilitated by networking/service computers (indicated as 2-132 and 2-140 in FIG. 2-1). Such computers may provide improved communication with devices that do not have their own networking and communication capabilities. Thus, the networking/service computers emulate the desired networking and communication functionality for their respective devices.

FIG. 2-2 illustrates one possible routing of communication between a client 2-240 and a plurality of exemplary devices (2-210, 2-215, 2-220, 2-225, and 2-230). The client 2-240 entity may include the management system described above in reference to FIG. 2-1. In the exemplary communication scheme of FIG. 2-2, an instrument directory 2-245 provides the client 2-240 with information about the devices. Then, the communication between the client 2-240 and the device(s) is facilitated via a messaging service 2-205. The messaging service 2-205 may rely on an instrument-type service provider 2-250 to format the communication according to the devices’ requirements.

FIGS. 2-3A and 2-3B illustrate two possible modes of communication between the various components. In a point-to-point method, a communication between a client and a device is performed in a generally chained manner. In an exemplary chain, a message from the client goes through a messaging service, through an instrument software, and then to the destination device (instrument hardware in FIG. 2-3A).

FIG. 2-3B illustrates a publish/subscribe messaging method. In an exemplary communication where the device wants to send a message to the client, the device sends a status signal to the service provider that in response publishes that status (along with an appropriate topic) to the messaging service. The messaging service does not automatically relay the status message to the client. Instead, the client subscribes to the messaging service with an appropriate topic. The messaging service matches the topic from the instrument software with that from the client, and sends the corresponding status message to the client.

FIG. 2-4 illustrates an exemplary instrument directory 2-405. Such a directory may include an instrument information 2-425 that contains information such as instrument name, physical location, type, group, publishing information, and other information useful for management of the instrument (device). The directory 2-405 may also include an instrument type provider information 2-440 that contains information about the instrument-type service provider described above in reference to FIGS. 2-1 and 2-2. The directory 2-405 may also include a list 2-430 of active instruments. The directory 2-405 may also include a storage device for storing information that facilitates various functions of the directory. The directory 2-405 is also shown to be connected to a network 2-415 so as to allow communication with other components.

FIG. 2-5 illustrates a possible functional relationship between an instrument hardware 2-510 and an instrument service provider 2-505 (described above in reference to FIG. 2-1). The hardware is depicted to be in communication with a networking and service middleware 2-515. Such middleware may be an integral part of the instrument, or may be an emulating component (networking and service computer in FIG. 2-1). The middleware 2-515 is in communication with other networked components via a network 2-508. The instrument service provider 2-505, also networked with the middleware 2-515, may include a plurality of functionalities, including a state model component 2-530 and a hardware commands component 2-540. Various aspects of the state model component and the hardware command component are described below in greater detail.

From the brief description of the integration framework in reference to FIGS. 2-1 to 2-5, it will be appreciated that the present teachings may be implemented in such a framework in a number of ways. For example, entities described herein as “translators” may be incorporated as part of the instrument service provider (FIG. 2-5). Similarly, entities described herein as “translations” may be part of the instrument interface (FIG. 2-4). It should be understood, however, that the various aspects of the present teachings may be implemented in systems other than that of the aforementioned framework without departing from the spirit of the present teachings.

As described above, the instruments can be advantageously integrated and recognized via an instrument directory. As will be described below, such a directory having information about the instruments allows the interface 1032 in FIG. 1) to advantageously provide an adaptable GUI for a given instrument.

FIG. 3 illustrates how the user interface 1032 can be constructed to provide access to a variety of assay-related monitoring information and/or controls by including an instrument-interface capability, along with similarly common interfaces for samples, analysis, and other study-related features. The exemplary monitor and/or control features are depicted as being categorized as either relating to laboratory management functions 1042 or to study management functions 1044.

The laboratory management functions 1042 may comprise an instrument management function 1046 and a sample management function 1050. The instrument management function 1046 may comprise a monitor/control function 1060, a diagnostic function 1062, and a calibration function 1064. The sample management function 1050 may comprise an inventory function 1066 and a sample tracking function 1070.

The study management function 1044 may comprise a run planning function 1052, a data run function 1054, and an analysis function 1056. The run planning function 1052 may comprise a standard run function 1072 and a customized run function 1074. The data run function 1054 may comprise a monitoring function 1076 and an online analysis function 1080. The analysis function 1056 may comprise a detailed analysis function 1082 and a simulation function 1084.
It will be appreciated that the aforementioned exemplary functionalities of an assay may be represented to the user in the form of a GUI for informational and/or control purposes. By having the details of the underlying features such as instruments, samples, run planning, and analysis stored in accessible manner at some functional location, the application that generates the GUIs can be configured to generate a GUI that adapts and populates itself according to the specifics of a given feature. It will be appreciated that such a commonality of user interface improves the manner in which the biological laboratory can be managed in general.

In the description below, the common GUI adapting to a variety of underlying features is described in terms of instruments. It will be understood, however, that similar concepts may be implemented for managing samples and other assay-related features without departing from the spirit of the present teachings.

FIG. 4A now illustrates a GUI application 1090 configured to generate a GUI. In one aspect of the present teachings, the application 1090 is configured to generate a common GUI that can be populated with particulars depending on the details of an instrument being interfaced with. Such a common GUI may be made accessible for the application 1090 to populate appropriately to generate the desired GUI.

Thus, the application 1090 generates a GUI 1092 for an instrument X 1104 based on available information 1100 for the instrument X. Similarly, the application 1090 generates a GUI 1094 for an instrument Y 1106 based on available information 1102 for the instrument Y. Such one-to-one relationship between a GUI and an instrument may be extended to any number of instruments.

As shown in FIG. 4A, the information 1100, 1102 about the exemplary instruments X and Y 1104, 1106 may be functionally located in a database 1096 that includes instrument information 1096. Such instrument information database is described below in greater detail in the form of an example.

As further shown in FIG. 4A, the application 1090 obtains the information 1100, 1102 for the exemplary instruments X and Y 1104, 1106 and generates the GUIs 1092, 1094 for X and Y, thereby allowing the user to monitor and/or control the instruments X and Y 1104, 1106. In certain embodiments, the interaction between the GUIs and the instruments may be performed via the application 1090. Alternatively, the GUIs may be configured to bypass the application and interact directly with the instruments. In either of these two alternatives, the interaction may be either directly or via a controller 1108.

FIG. 4B illustrates that in certain embodiments, a user interface application 1214 may be configured to launch a web browser 1226. Such interfaces may comprise a GUI 1224, a web browser based user interface 1226, or any combination thereof. The use of web browsers as a basis for the user interface may be advantageous in certain platforms that support the user interface application 1214. In certain embodiments, such web browsers may be implemented using the HTML code known in the art.

The user interface application 1214 may populate the GUI 1224 and/or the web browser 1226 using information obtained from an information source such as an instrument directory 1210 in a manner similar to that described above in reference to FIG. 4A. The directory 1210 is depicted to include available information 1212 for a plurality of exemplary instruments 1216 to 1220, etc. Also similar to the configuration of FIG. 4A, the interaction between the instruments 1216 and the user interface application 1214 may be either direct or via a controller 1220.

FIG. 4C illustrates a system 1240 of certain embodiments configured to dynamically generate a GUI 1254 when an exemplary instrument (denoted as “YY”) 1242 is added to the system 1240. A directory 1246 may be updated with information 1250 about the instrument “YY” 1242 in a manner described above. Such updating of the directory 1246 may be facilitated by a management system 1244.

The addition of the instrument “YY” 1242 to the system 1240 may cause the management system 1244 to discover the instrument “YY” 1242 in a variety of ways. For example, the management system 1244 may routinely check the directory 1246 and note the additional entry. In another example, the addition of the instrument’s information 1250 in the directory may be accompanied by a broadcast to the system 1240, informing the system 1240 of the addition.

The management system 1244 may be configured such that upon such detection of an added instrument, it may invoke a GUI application 1252 to launch a GUI 1254. The GUI 1254 generated in such a manner may include the instrument’s available protocol as well as other features that facilitate the sample assay process. The GUI application 1252 may be configured to generate the GUI 1254 by populating a common template GUI, based on information 1250 for the instrument “YY” 1242, in a manner similar to that described above in reference to FIG. 4A. Thus, one can see that such a functionality allows a user to add a compatible instrument to the system 1240 and be provided with a visual interface that aids in incorporating the added instrument into an assay process.

FIGS. 5A-C illustrate an exemplary instrument information 1112 and its corresponding GUIs 1114 and 1120 that may be generated by the application 1090 described above. The exemplary instrument information 1112 may be part of an instrument directory 1110. The instrument directory is described above in reference to the system framework.

As shown in the instrument information 1112, the exemplary information may comprise an instrument identifier. In the exemplary information, the instrument is designated as “A” that represents an array device. The information further comprises available monitor parameters and available control parameters for the array device.

The exemplary available monitor device may comprise the parameters listed in FIG. 5A. As shown in FIG. 5B, such parameters populate the GUI 1114 by the application 1090. In certain embodiments, the order of parameters in the information 1112 may determine the manner in which they are arranged on the GUI 1114. Additional sub-parameters associated with the parameters may also determine how the parameters are displayed on the GUI 1114. For example, exemplary parameters numbered as 15 to 20 have associated with them a series of input values to be inter-
interpreted in the following exemplary manner: first sub-parameter is the unit; second and third sub-parameters are the lower and upper limits for actual value (fourth sub-parameter) and set/nominal value (fifth sub-parameter). Thus, for the exemplary parameter “EP Voltage,” the unit to be displayed is “KV,” and the lower and upper limits of a scale to be displayed are 0 and 15 KV, respectively. The fourth sub-parameter of “EP_V_actual” is a value of the actual voltage of instrument “A” 1116 (1.0 KV in the exemplary GUI of FIG. 5B). The fifth sub-parameter of “EP_V_set” is a value of the set voltage for the instrument 1116 (3.0 KV in the GUI of FIG. 5B).

[0075] The ranges of parameter values in the foregoing example may be formatted so as to instruct the application to display them in a certain manner. It will be appreciated that there are many number of ways information can be displayed on a GUI. It will be further appreciated that the information for instruments may be formatted in any number of ways to provide any number of display formats of the GUIs without departing from the spirit of the present teachings.

[0076] FIG. 5C illustrates an exemplary GUI 1120 launched by the application 1090 based on the information 1112. The exemplary available control parameters may comprise a laser operation and a comment function. The laser function may be formatted in the following exemplary manner: first sub-parameter is the unit; second and third sub-parameters are the lower and upper limits to be displayed proximate a fourth sub-parameter of a user input value. The laser function may comprise one of the following exemplary choices: “Sci Power”, “Apply Power”, and “Shut Down.” As previously described, in certain embodiments, the manner in which the control parameters are displayed may be determined by the format of the listed control parameters. In the exemplary GUI 1120 described above, the one or more control commands selected through the GUI may be sent to the exemplary instrument 1116.

[0077] FIG. 6 now illustrates an exemplary GUI 1130 launched by the application 1090 based on a sample list 1132. The sample list 1132 could be considered to be a similar “information” about the samples analogous to the instrument information described above. The sample list 1132 could contain information about the inventory of substantially all samples within the system, or about the whereabouts of a particular sample. The sample list 1132 could be facilitated by the sample-identifying means such as the barcode 1006 described above in reference to FIG. 1.

[0078] FIGS. 7A-C now illustrates some exemplary GUIs that may be launched by the application 1090 based on some combination of instrument information and sample information. Such combination of information can facilitate the various laboratory and study management functions described above in reference to FIG. 3.

[0079] FIG. 7A illustrates an exemplary GUI 1140 that may be launched by the application 1090 based on the collection of instruments’ information (1096 in FIG. 4A). In particular, the exemplary GUI 1140 is configured to list instruments that may be used with a particular type of an exemplary activity (“Aliquot DNA”) with a particular type of an exemplary assay process (“ProcessRun02”). As shown in FIG. 7A, the GUI 1140 can also list instruments associated with other exemplary activities “Sample Setup” and “Perform PCR,” with their respective exemplary processes “ProcessRun01” and “ProcessRun03.”

[0080] FIG. 7B illustrates an exemplary GUI 1142 that may be launched by the application 1090 based on information provided by an activity manager 1144. The activity manager 1144 may be a process or a database that combines information from the sample list 1132 and the instrument information 1096. The exemplary GUI 1142 lists the samples (“C1234567890” that are to be processed for the exemplary activity “Aliquot DNA” via the exemplary process “ProcessRun02.”

[0081] FIG. 7C illustrates an exemplary GUI 1146 that may be launched by the application 1090 based on information provided by a run scheduler 1150. The run scheduler 1150 may be a process or a database that combines information from the sample list 1132 and the instrument information 1096 to schedule processing of the samples via the available instruments for one or more types of data-taking runs. As shown in FIG. 7C, the particular GUI 1146 lists runs to be performed by an exemplary instrument “3100test1.”

[0082] It will be appreciated that the various exemplary GUIs described above in reference to FIGS. 5-7 can be considered to be various populated form of a single GUI launched by the application 1090. The various populations are based on the instrument and/or sample information stored in a database separate from the application that provides the interface with the user. The information provided to the application may be processed by some intermediate process or database such as the exemplary activity manager 1144 in FIG. 7B and the run scheduler 1150 in FIG. 7C. In either of these two cases, however, the information provided to the GUI application 1090 is generally stored separately from the application 1090, and can be considered to be another form of database on which the GUIs are based upon.

[0083] This concept of providing the GUI application with information for the adaptive populating GUI can be extended to data presentation. FIG. 8A illustrates an exemplary GUI 1160 displaying a raw data from an instrument “my3100.” The exemplary GUI 1160 may be launched by the application 1090 based on the instrument information 1096 that corresponds to an instrument 1162. In particular, the instrument 1162 comprises the instrument “my3100.”

[0084] The raw data displaying GUI 1160 displays the data as it is being output from the instrument 1162. As previously described, the manner in which the GUI display a given parameter could be determined by the format of the parameter in the instrument’s information. Thus, the exemplary raw data parameter could be formatted so as to make the application 1096 interpret the raw data output from the instrument 1162 as a streaming graph-format display.

[0085] FIG. 8B illustrates another exemplary data-displaying GUI 1170 that may be launched by the application 1090 based on the data from an exemplary instrument “kiaowo” 1172. Information about such an instrument may be obtained from the instrument information 1096. The information about the instrument 1172 may include an output data format that allows the GUI 1170 to display the data in a two-dimensional “Gel display” format.

[0086] FIG. 9 illustrates another exemplary data-displaying GUI 1180 that may be launched by the application 1090...
based on the data output from an analysis module 1182. In certain embodiments, the analysis modules that process the data from the instruments could be treated like the instruments in terms of integration and being listed in some form of a directory. That is, the analysis modules could be listed in a database similar to the instrument directory described above. Such a database could also list each analysis module’s available outputs.

[0087] Thus, the exemplary GUI 1180 displays a post-instrument(s) processed data from the analysis module 1182. The analysis module 1182 can also output the processed data to a result database 1184 for later GUI display and/or other form or presentation.

[0088] The various exemplary GUIs described above can be generalized as being a single or a few GUIs that can be populated by information obtained from some source accessible by the application that generates the GUI. In certain embodiments, the application generates a single GUI that adapts to the various information.

[0089] FIG. 10 now illustrates a process 1190 for generating the aforementioned GUIs. The process 1190 may be performed by the GUI application 1090. The process 1190 begins as a start step 1192, and in step 1194 that follows, the process 1190 receives a request for a user interface. Such a request may originate from an existing GUI. For example, a user may request additional information about the instrument management (1046 in FIG. 3) from an existing GUI that displays the lab management summary 1042. Similarly, the user may request additional information on a particular instrument from the thus-created GUI (of instrument management 1046), thereby creating the monitor/control functionality 1060 GUI.

[0090] In step 1196 that follows, the process 1190 obtains information associated with the user interface. In step 1200 that follows, the process 1190 generates a GUI populated with the information obtained. The process 1190 ends at a stop step 1202.

[0091] Control And Management Services

[0092] In one aspect, the present teachings define a set of services (e.g., instrument and applications services) that are commonly supported within the LIMS system. These services desirably aid in control and management simplification across different instrument and application platforms and facilitate integration of these components into the unified framework of the LIMS system.

[0093] In various embodiments, the service sets provide means for control and monitoring of various components, including instruments and applications, within the LIMS environment. Additionally, these service sets may be desirably configured to enable remote connectivity via a network or other communication means. In general, components of the LIMS system may desirably implement a standardized instrument or application service interface thereby providing a consistent set of interfaces which facilitate component integration into the LIMS system. In instances where an interface may not be applicable to a certain component, instrument, or application type, the component may recognize and indicate that the interface is not applicable.

[0094] In one aspect, the design and implementation of the component application programming interface(s) (API) provide a framework of commonality in which a user interface (GUI) may be implemented on the basis of an instrument services contract. Implementation in this manner may be substantially independent of any instrument specific while still being able to control and monitor a selected component configured to utilize the service set.

[0095] The following description details various aspects of the service sets and provides examples of how these services may be implemented in the LIMS system. These functionalities are meant to illustrate possible configurations which may be utilized by the LIMS system and should not be construed to limit its functionality.

[0096] In one aspect, a service set may comprise a dynamic container creation service. FIG. 11 illustrates the communication sequence and functionalities of the dynamic container creation service 1500. In one aspect the dynamic container creation service 1500 functions to provide application dependent container type information. The type of information that is gathered for each container typically depends on analysis application or instrument. In one exemplary application, when there was a new type of analysis application is made available which is to be desirably integrated into the system, other applications, such as data collection software, may need to be adjusted or modified to support the new analysis application. One feature of the dynamic container creation service 1500 is that it provides support for “plug-in” integration such that client applications and instruments can dynamically gather information from a container type service provider, such as an analysis application, and dynamically construct a container editor 1505.

[0097] In one aspect each instrument services service provider may provide an application installer 1510 that registers a selected application for a particular instrument type. However, the service provider need not be the sole provider for this information. As an example, a selected instrument (e.g. sequencer) may populate a set of applications (e.g., a default fragment analysis application). If a new analysis application is created for the instrument platform, an application installer 1510 can be provided that utilizes the existing set of attributes independent of the instrument itself.

[0098] In one aspect, the instrument services provide an interface implementation for an application installer to register application specific container 1520 and sample definitions, an example of which is shown below:

[0099] ICFResult ICFRegisterContainerParameters("container definition") 1530;

[0100] ICFResult ICFRegisterApplicationParameters("application definition") 1540;

[0101] In this example, the parameters, containerDefinition and applicationDefinition, may be implemented in XML format and can be validated by a suitable XML schema (see FIGS. 16-17). The returned ICFResult object contains ICFResult.RESULT_CODE_OK in ICFResult.getResultCode() upon successful operation.

[0102] In another aspect, instrument services may also provide an interface implementation for a type service provider to retrieve specific container and application parameters, an example of which is shown below:

[0103] ICFResult ICFGetContainerParameters() 1550;

[0104] ICFResult ICFGetApplicationParameters() 1560;

[0105] In this example, if the operation is determined to be successful, the returned ICFResult object may be designated to contain an XML string definition that was installed by the
application installer using the ICFRegisterApplication call. A successful operation results in the returned ICFResult object containing an XML sample definition. The type and content of the information provided depends on the provider's implementation and it will be appreciated that a type service provider may be part of an instrument service provider or a standalone type service provider.

[0106] In another aspect, the GUI client 1505 may use the XML container definition to construct a container creator. FIG. 12 illustrates a sample snapshot 1600 from a data collection instrument. In this example, the GUI container creator may comprise attributes, such as “Container Name”, “Plate ID”, “Description”, among others. These attributes may be extracted from the XML container definition registered by the application installer. In one aspect, a customized container editor for “Genemapper” may be constructed following execution of the “OK” button 1605. In this example the customized container editor may extract column names from an XML sample definition registered by application installer.

[0107] In another aspect, container import may be implemented as functions commonly used in instruments supported by instrument services. The following FIGS. 13-15 illustrate the “pulling” and “pushing” of a logical container to import the container into a instrument service provider 1710. As shown in FIG. 13, a custom client 1705 may use the function subscribeInstrumentEvents() to prepare itself to receive information from the ICFStatusEvents() and ICFContainerStatusEvents() functions. Once ready, a custom client 1705 may then use ICFRegisterContainerProvider() call 1715 to notify the instrument service provider (ISP) 1720 that the custom client 1705 is available to provide the desired container information. This function call may further return a Boolean result to indicate an ISP acknowledgment and further indicate if the container import functionality is supported. The client 1705 may further use this result code to determine if the ISP 1720 supports the selected feature. Subsequently, the ISP 1720 may notify the Instrument to process the next available container upon receiving a “start run” command from the user interface component (not shown in Figure). In various embodiments, the ISP 1720 may receive a container barcode identifier as the return of processNextContainer( ) 1725. The ISP may then attempt to locate the container’s information from its own database. The flowchart shown in FIG. 14 illustrates details for this conditional situation.

[0108] In one aspect, if the desired information 1825 is available from the database 1830, the ISP 1720 may schedule a run 1835 for this container and perform run(s) using this container. If ISP 1720 does not find the container’s information in its database and at least one custom client has registered with the ISP 1720 to provide container information, the ISP 1720 may send out a ICFContainerStatusEvents( ) 1840 call to the messaging service. A custom client 1705 receives this event and uses the importContainer( ) function to provide container information to the ISP 1720. When the ISP 1720 finishes the scheduled run(s) 1835 for this container, a “run completed” notification may be sent in the ICFStatusEvents() function. Subsequently, the custom client 1705 may update the status for the container in its database.

[0109] FIG. 15 illustrates how a custom client 1705 can “push” container information to a service provider 1720 to persist in the service provider’s database allowing it to be run at a later time. In the example, the custom client 1720, which may be a GUI that allows user to type or scan a container barcode or other identifier sample information, requests container information from the LIMS system 1930. Once the container information is acquired, the client 1705 may use the importContainer( ) function 1940 to “push” it into ISP 1720.

[0110] In one aspect, the service provider 1720 provides logical container information by implementing the aforementioned interface definition to provide synchronous service. This service is designed to provide some of the functionalities illustrated in the use case diagram (FIG. 15) for integrating with a LIMS client 1930. In various embodiments, an instrument service provider 1720 may implement the interface according to the function:

[0111] ICFResult importContainer(String plate Text);

[0112] Here the client may provide an XML data string (conforming to the container XML schema) as a parameter for the call. An ICFResult object containing ICFResult.RESULT_CODE_OK in ICFResult.getCode() may then be returned upon a successful operation. The service provider may further publish an ICFContainerStatusEvents function when container information is desirable to run or execute a physical container.

[0113] FIGS. 16-20 illustrate various embodiments of XML schema for application and container definitions that may be used in implementation of the aforementioned service sets and container creation/utilization services. It will be appreciated by one of skill in the art that these schemas may be readily modified to accommodate new or different data types without departing from the scope of the present teachings.

[0114] The aforementioned description of instrument and application services desirably provides a highly configurable and readily modifiable means for control and management across different instrument and application platforms. Furthermore, these services provide for improved integration of sample identification functionality requiring little or no user input.

[0115] In one aspect, the “push”/“pull” functionality described above provides a convenient method for implementing “just in time” sample identification and information import functionality wherein a selected instrument can obtain information for a particular sample or analysis on an as needed basis rather than storing a large amount of data describing numerous samples which may or may not be utilized. The just in time functionality helps to provide appropriate sample and analysis information on an as need basis thereby conserving system resources and providing for relatively seamless integration of instruments and applications within the LIMS environment. Additional details of the just in time functionality are described elsewhere.

[0116] Dynamic Loading of User Interface

[0117] In various embodiments, the user interface may be desirably implemented in a dynamically loaded manner such that a relatively consistent view is maintained for different instrument and application types. View consistency typically improves the user’s ability to interact with the LIMS system through the user interface as the user may become accus-
tomed to a generic application and instrument presentation wherein fields, buttons, and other information is situated in the user interface in a predicative and uniform manner.

Additionally, in conventional systems changes in an application or instrument may affect the user interface and reconfiguration of the user interface can take up much time and effort. Modifying or adding instruments or applications may significantly affect a client side viewing or interaction application and necessitate recompiling, testing, and shipping of a new/upated vendor side application which must be reinstalled on the client side.

In various embodiments, to avoid such costly and time consuming procedures, the present teachings provide means for dynamically loading of classes and components for instruments and applications. In one aspect, the client side user interface may utilize a dynamic loading functionality implemented in a programming language such as JAVA wherein various instrument and application classes are contained in jars. Additionally, it will be appreciated that the dynamic loading prototype may be used in other areas of the LIMS system outside of the user interface to provide a similar dynamic loading functionality.

In one exemplary implementation of a dynamically loaded user interface using the JAVA language, an application may be contained in a frame (JFrame) with a pane (Panel) contained in the frames. A panel is a view of the application and generally comprises the expected components for a selected view. In implementing the user interface, a plurality of panels may be selected from a menu list. Typically, when a user launches the user interface application, a frame may be displayed. The user them may select an application menu item and an associated panel corresponding to the desired view will be displayed. The panel will generally contain all the necessary components for that view and correspond for example to a selected instrument or data analysis application.

FIG. 21 illustrates one method 4500 for implementing a web-browser based dynamically loaded user interface implemented using the JAVA language. In state 4505, when a panel displayed in the user interface is selected, an associated panel name may be retrieved in state 4510. This name is used as an identifier to identify the appropriate jar file associated with it. For example, if a particular panel named “Run3100” is selected, a corresponding file “Run3100.jar” will be looked up in state 4515. Typically, application jars may be stored locally in a directory on the client side computer. Furthermore, a time stamp may be associated with the file which is retrieved from the file upon lookup in state 4520. This time stamp may be compared in state 4525 to that of a remotely maintained file stored on a server computer (web server). If the file does not exist or if the times stamp for the file on the client computer is outdated with respect to the remotely maintained file, the client may retrieve a new/updated file from the server computer in state 4530. Once the file has been updated, or if the file on the client computer is not outdated, the file may be executed in state 4535 and the user interface will display the appropriate information to the user via the panel.

In various embodiments, the client/server communications and interactions may occur in a web/browser based environment. Thus the client computer may implement the user interface through a web-browser and a connection to a web server may be established using an HTTP protocol using the client’s IP address as run time input. The time stamp of the panel’s jar file may then be retrieved and compared with the local file’s time. If the server’s file is newer, it may cache/replace the old file with the new. In the final stage of user interface loading a class loader may be called to load the classes needed in the panel to thereby populate the panel with the appropriate information to be displayed to the user.

It will be appreciated that the aforementioned implementation of a dynamically generated user interface represent but one of many possible implementations the may be achieved through application of the present teachings. As such other implementations and constructions of a dynamically generated user interface are considered but other embodiments of the present teachings. Furthermore, it will be appreciated that other programming languages may be used as a means to implement the user interface and the user interface may further be constructed as a standalone application without the requirement of a web browser.

Although the above-disclosed embodiments of the present invention have shown, described, and pointed out the fundamental novel features of the invention as applied to the above-disclosed embodiments, it should be understood that various omissions, substitutions, and changes in the form of the detail of the devices, systems, and/or methods illustrated may be made by those skilled in the art without departing from the scope of the present invention. Consequently, the scope of the invention should not be limited to the foregoing description, but should be defined by the appended claims.

All publications and patent applications mentioned in this specification are indicative of the level of skill of those skilled in the art to which this invention pertains. All publications and patent applications are herein incorporated by reference to the same extent as if each individual publication or patent application was specifically and individually indicated to be incorporated by reference.

What is claimed is:

1. A system for providing a graphic user interface for a biological laboratory having a plurality of biological processing instruments adapted to process a plurality of biological samples, the system comprising:

   an information component for instruments that includes information about the plurality of the biological processing instruments, wherein the information for a given instrument comprises a list of the instrument’s parameters that can be monitored and a list of the instrument’s parameters that can be controlled; and

   an interface application that generates the graphic user interface based on the information for one or more of the instruments wherein the interface thus generated is populated according to at least a portion of the instrument’s monitor and control parameters thereby allowing the interface application to generate different interfaces based on a single adaptable template interface.

2. The system of claim 1, wherein the graphic user interface for a given instrument allows a user to monitor and control the instrument.

3. The system of claim 1, wherein the interface application uses the information about the plurality of instruments
to generate an instrument management graphic user interface that lists the plurality of instruments available for use.

4. The system of claim 1, further comprising an information component for samples that includes information about the plurality of biological samples wherein the samples' information component allows the interface application to generate adaptable graphic user interfaces based on the samples' information.

5. The system of claim 4, wherein the interface application uses the information about the plurality of biological samples to generate a sample management graphic user interface that lists the samples in the biological laboratory.

6. The system of claim 5, wherein the sample and instrument information are combined to allow the interface application to generate a study management graphic user interface.

7. The system of claim 6, wherein the graphic user interface is adapted to allow a user to plan a manner in which data is taken during a run.

8. The system of claim 1, further comprising an information component for an analysis module that analyzes the data from one or more biological processing instruments wherein the analysis module's information component allows the interface application to generate adaptable graphic user interfaces based for viewing results of the analyzed data.

9. The system of claim 1, wherein the plurality of biological processing instruments includes a sample storage device.

10. The system of claim 1, wherein the plurality of biological processing instruments includes a sample transferring robotics device.

11. The system of claim 1, wherein the plurality of biological processing instruments includes a thermal cycler device.

12. The system of claim 1, wherein the plurality of biological processing instruments includes a mass spectrometer.

13. The system of claim 1, wherein the plurality of biological processing instruments includes a sequencer device.

14. The system of claim 1, wherein the plurality of biological processing instruments includes an electrophoresis device.

15. The system of claim 1, wherein the plurality of biological processing instruments includes an array device.

16. The system of claim 1, wherein the plurality of biological processing instruments includes an analysis computing device.

17. The system of claim 1, wherein the plurality of biological processing instruments includes a data storage device.

18. The system of claim 1, wherein the instrument component is part of an instrument directory.

19. The system of claim 1, wherein the graphic user interface is based on a web browser.

20. The system of claim 1, wherein the interface application is dynamically induced to generate a graphic user interface corresponding to an added instrument.

21. The system of claim 20, wherein the dynamically generated graphic user interface corresponding to the added instrument includes one or more protocols that can be provided by the added instrument.

22. A method of generating a graphic user interface for a biological laboratory having a plurality of biological processing instruments adapted to process a plurality of biological samples, the method comprising:

- receiving a request for a graphic user interface;
- obtaining information associated with the requested graphic user interface; and
- generating the requested graphic user interface by populating a template graphic user interface based on the information obtained.

23. The method of claim 22, wherein receiving the request comprises receiving a request for a graphic user interface for one or more biological processing instruments.

24. The method of claim 23, wherein obtaining information comprises obtaining information about the one or more biological processing instruments from an instrument directory.

25. The method of claim 22, wherein receiving the request comprises receiving a request for a graphic user interface for one or more biological samples.

26. The method of claim 25, wherein obtaining information comprises obtaining information about the one or more biological samples from a sample list.

27. A control system for a biological laboratory having a centralized user interface that provides monitoring and coordination functionality directed towards the operation of a plurality of laboratory components, the centralized user interface comprising:

- an instrument information acquisition component that acquires information about the plurality of laboratory components, wherein the information for a selected laboratory component comprises monitoring and control parameters; and
- an instrument interface generation component that generates a user interface using an adaptable interface template that is populated with at least a portion of the information acquired by the instrument information acquisition component so as to provide a means for selectively generating laboratory component interfaces using the information associated with one or more of the laboratory components.

28. The system of claim 27, wherein the adaptable interface template used to generate the user interface receives the information for a selected laboratory component and automatically associates one or more presentation fields with information directed towards instrument monitoring.

29. The system of claim 27, wherein the adaptable interface template used to generate the user interface receives the information for a selected laboratory component and automatically determines the current state of operation for one or more selected instruments which is thereafter integrated into the user interface.

30. The system of claim 27, wherein the instrument interface generation component receives the information from the instrument information acquisition component and determines the availability of one or more selected instruments which is thereafter integrated into the user interface.

31. The system of claim 27, wherein the instrument interface generation component receives the information from the instrument information acquisition component and determines the availability of one or more selected instruments which is thereby integrated into the user interface.
from the instrument information acquisition component and determines a current sample or data set operated on by one or more selected instruments which is thereafter integrated into the user interface.

33. The system of claim 27, wherein the instrument information acquisition component further provides means for determining a current sample inventory which is thereafter integrated into the user interface by the instrument interface generation component.

34. The system of claim 27, wherein the user interface is based on a web browser.

35. The system of claim 27, wherein the instrument interface generation component is dynamically induced to generate a user interface corresponding to an added instrument.

36. The system of claim 35, wherein the dynamically generated user interface corresponding to the added instrument includes one or more protocols that can be provided by the added instrument.